HANDLING LOCAL MACHINE CHECK EXCEPTIONS IN LINUX*

BY **ASHOK RAJ** ON DEC 27, 2017

INTRODUCTION

Mission critical servers must avoid downtime. Systems that require high availability as a part of their Service Level Agreement (SLA) often require 24x7 uptime or 99.999% availability (also called 5-9's). This translates to about five minutes of total downtime each year. IT installations require high availability because the loss of business caused by downtime could be very expensive. These installations typically have several servers with fault tolerance, such as backup servers. An example is the Stratus* computer that has been running for 24 years, and hasn't quit yet.[i]

One way to reduce downtime is to improve how the system handles hardware exceptions. The Intel® Xeon® family of processors notifies the OS about hardware exceptions by signaling Machine Check Exceptions (MCEs). MCEs are broadcast to all CPUs in the Intel® processor family. This was designed to limit exposure to uncorrectable errors (UC), such as the case where the data is sent to a requester and the signaling of UC is still stuck in the pipeline. By stopping the processing of all the logical CPUs in the system, the system could be stopped more quickly preventing consumption of the corrupt data. For a full description of all the MCE features and OS recommended actions, refer to Chapter 15 Machine Check Architecture, Volume 3B of the Intel® 64 and IA-32 Architectures Software Developer's Manual (SDM).

The Intel® Xeon® family of processors, starting with the X65xx and X75xx generation, have supported recovery from certain uncorrectable errors (UE), specifically Patrol Scrub (PS) errors and Explicit Write Back (EWB) errors from Last level cache (LLC). These errors are referred to as uncorrected recoverable (UCR) errors; Linux attempts to recover from these UCR errors because they do not require a shutdown, as opposed to UC fatal errors. UCR errors were introduced starting with Intel® Xeon® processors based on Intel® microarchitecture, code named Nehalem. These errors are asynchronous and are not encountered in the execution path. The SDM refers to UCR errors as Software Recoverable Action Optional (SRAO), meaning no action is required at this time, because the corrupt data is not immediately being used. Therefore it is safe for the system to continue to run. Linux added support for recovery from SRAO errors around the v2.6.32 timeframe.

Intel further enhanced MCE handling, starting with the E5-16xx, 26xx, and 46xx series of Intel® Xeon® family of processors, by adding recovery from certain memory errors in the execution path (also known as memory poisoning support). In cases when the CPU and platform enable poisoning support, when UC errors are discovered, the data is tagged as poisoned and returned to the requester. When these errors are detected synchronously in the execution path, they are reported by the Instruction Fetch Unit (IFU) or Data Cache Unit (DCU). This class of errors are referred to as Software Recoverable Action Required (SRAR) to indicate that system software must perform some recovery before

with a poison indication, it prevented any potential data corruption scenario since the data was already tagged and any attempted use of poisoned data would have been signaled before consumption.

While this approach worked well for many generations, broadcasting errors when only a single executing thread is affected is very disruptive and can cause unintended fatal errors. For example, the Machine Check Architecture in the Intel® processor family does not permit nested machine checks. If a second MCE is signaled while a logical processor is still handling an earlier broadcast MCE, an automatic hardware-initiated shutdown will occur. Imagine a multi-threaded application reading shared data. If two or more threads read the same poisoned memory location, it would result in a system shutdown instead of allowing the system software to contain the failure.

Modern servers often host numerous virtual machines, each of which is supposed to be isolated from all others. However, modern Linux Virtualization uses Kernel Samepage Merging (KSM). KSM is a memory-saving de-duplication feature that merges anonymous (private) pages. While this technology reduces the overall memory footprint, it is another common shared memory scenario where a single memory fault could cause a nested MCE and bring down the entire system.

System architectures have evolved to support modern cloud and data-center needs. One such configuration is the use of Non-transparent bridge (NTB) that uses PCI to connect between an Intel® architecture system and other Intel architecture or non-Intel architecture systems. There are several applications for NTB; one such case uses the link for fast data duplication between systems. Advances in PCIe such as Enhanced Downstream Port Containment (eDPC) allow memory reads from PCIe to return poison. When multiple processors access this memory region, this could easily trigger several MCE's as a result of simultaneously accessing an affected region. If MCE's are continued to be broadcast, this crossfire could immediately result in system shutdown.

To accommodate these modern system architectures, Intel introduced the concept of Local Machine Check Exception (LMCE). LMCE was first introduced in Intel® Xeon® Scalable processors (formerly known as Skylake). Linux support for LMCE was added in v4.2

LOCAL MACHINE CHECK ARCHITECTURE

To accommodate legacy operating systems that always assume MCEs are broadcast, the default behavior of the hardware is to continue to broadcast MCEs. To avoid broadcast related issues, an enlightened OS could opt-in to LMCE, which would allow the system to behave in a more robust manner.

LMCE IDENTIFYING, ENABLING, AND DETECTION

The following section explains how to identify if an Intel® Xeon® processor supports Local Machine Check Exception (LMCE), identifying BIOS support for the feature, and how to inject and test if Linux recovered from an exception.

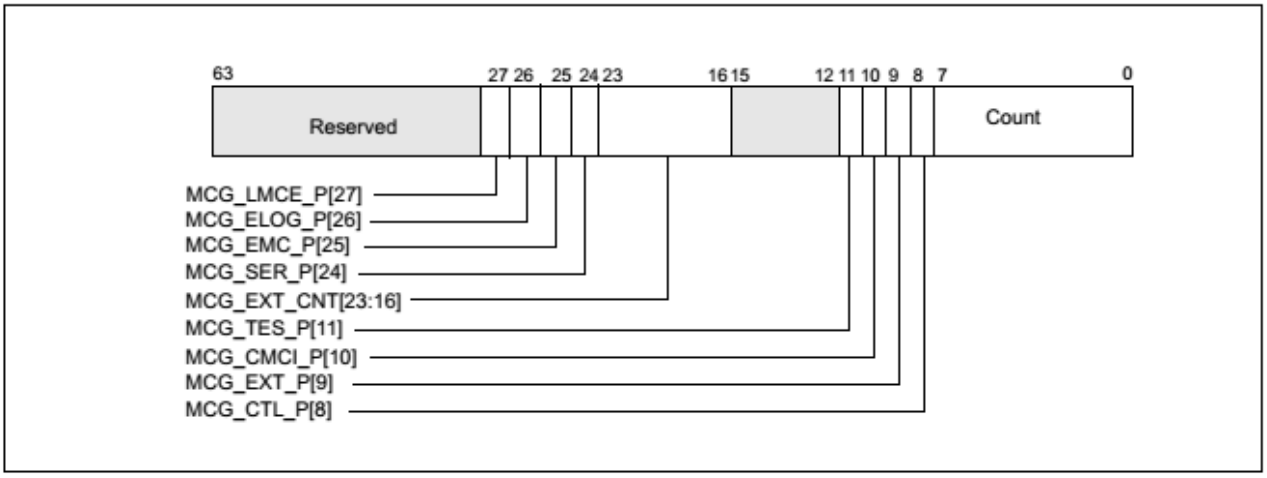


Figure 1: IA32_MCG_CAP[27] indicates presence of the feature.

To benefit from the LMCE-type signaling, OSes must opt-in to request hardware to avoid broadcasting these errors. To opt-in, the OS sets bit0 in MSR IA32_MCG_EXT_CTL to indicates that SRAR type errors can be signaled to only the affected logical CPU.

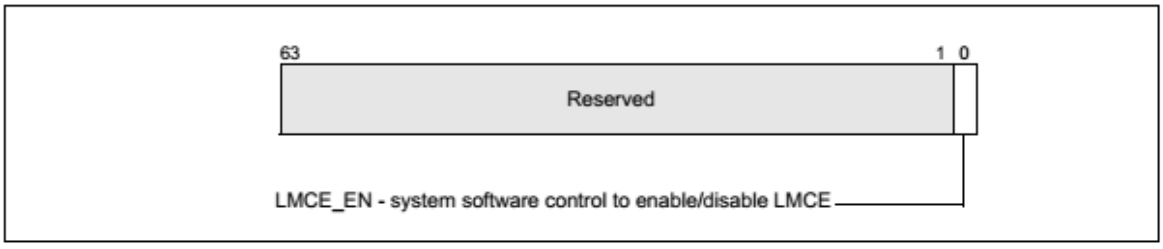
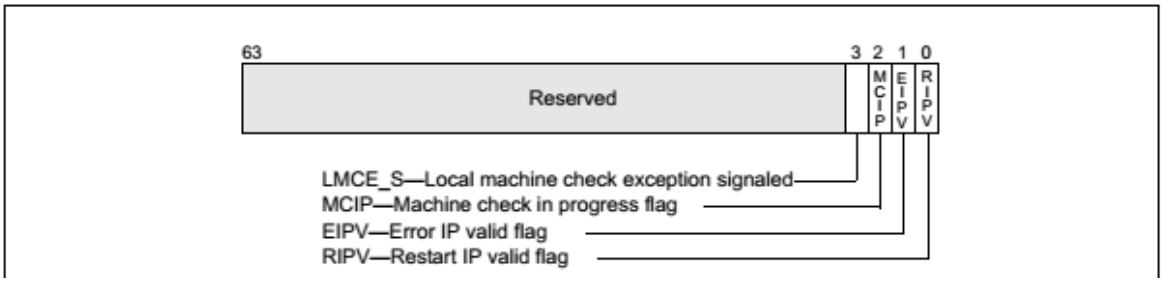


Figure 2: IA32_MCG_EXT_CTL - MSR to enable Local Machine Check



By default, Linux automatically enables LMCE if the CPU reports it and it is enabled in the BIOS. The BIOS configuration to enable MCE and LMCE is shown below.

HOW TO IDENTIFY IF LMCE IS ENABLED IN BIOS

1. Ensure that *msrtools* is installed before performing the next step.
2. Check if IA32_MCG_CAP (0x179) shows LMCE is reported as available in hardware.

```
#rdmsr 0x179
F000c14
Binary: 1111 0000 0000 0000 1100 0001 0100
MSR_IA32_MCG_CAP (0x179): bit 27  indicates LMCE is available.
```

3. Check that BIOS has enabled the feature. MSR_IA32_FEATURE_CONTROL (0x3a)

```
#rdmsr 0x3a
100005
Binary: 00010000 0000 0000 0000 0101
MSR_IA32_FEATURE_CONTROL.bit20 is set  indicates LMCE is enabled by BIOS for OS use.
```

4. Check if OS enabled the feature by checking MSR_IA32_EXT_MCG_CTL (0x4d0)

```
#rdmsr 0x4d0
1
```

/PROC/INTERRUPTS

One way you can verify that your Linux OS has enabled LMCE versus broadcast MCE's is via the */proc/interrupts* MCE entry. When LMCE is enabled and an MCE is signaled to only a single logical processor, */proc/interrupts* would show that the exception was taken only by one CPU as indicated below. When MCE is broadcast to all CPUs, all processors count the event.

```
#cat /proc/interrupts | grep MCE
MCE:       0    0    0    1
```

The example above shows one interrupt in a system that has four logical CPUs.

MCELOG

Another way to verify that your LInux OS has enabled LMCE is by checking output of the mcelog daemon. The MCG status line will show that LMCE is enabled. Sample output below.

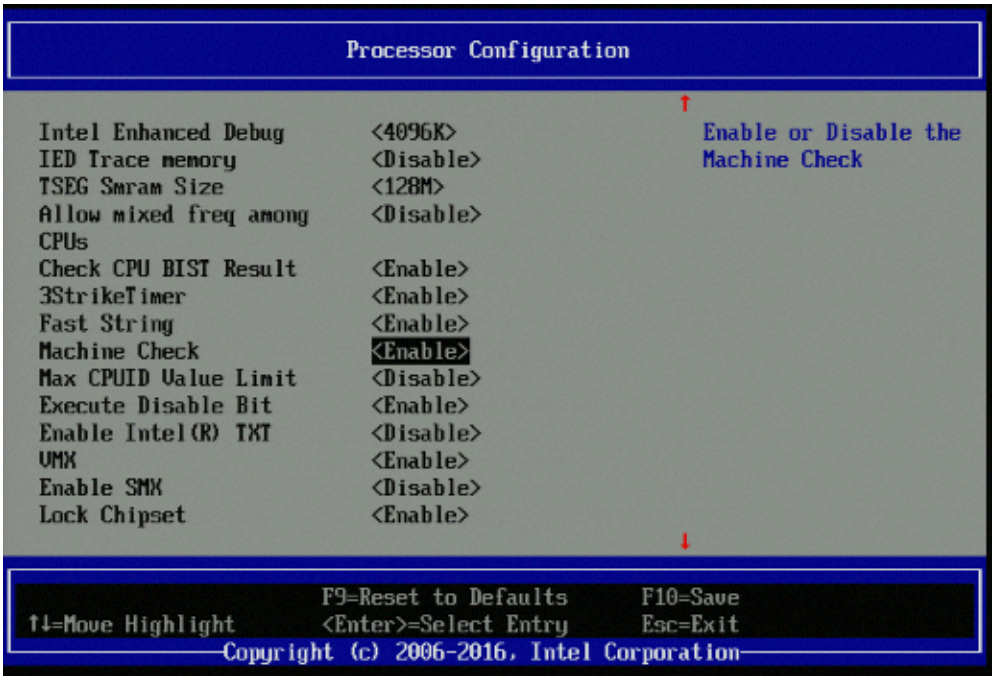
```
Uncorrected error
Error enabled
MCi_MISC register valid
MCi_ADDR register valid
SRAR
MCA: Data CACHE Level-0 Data-Read Error
STATUS bd80000000100134 MCGSTATUS f
MCGCAP f000c14 APICID e6 SOCKETID 3
CPUID Vendor Intel Family 6 Model 85
```

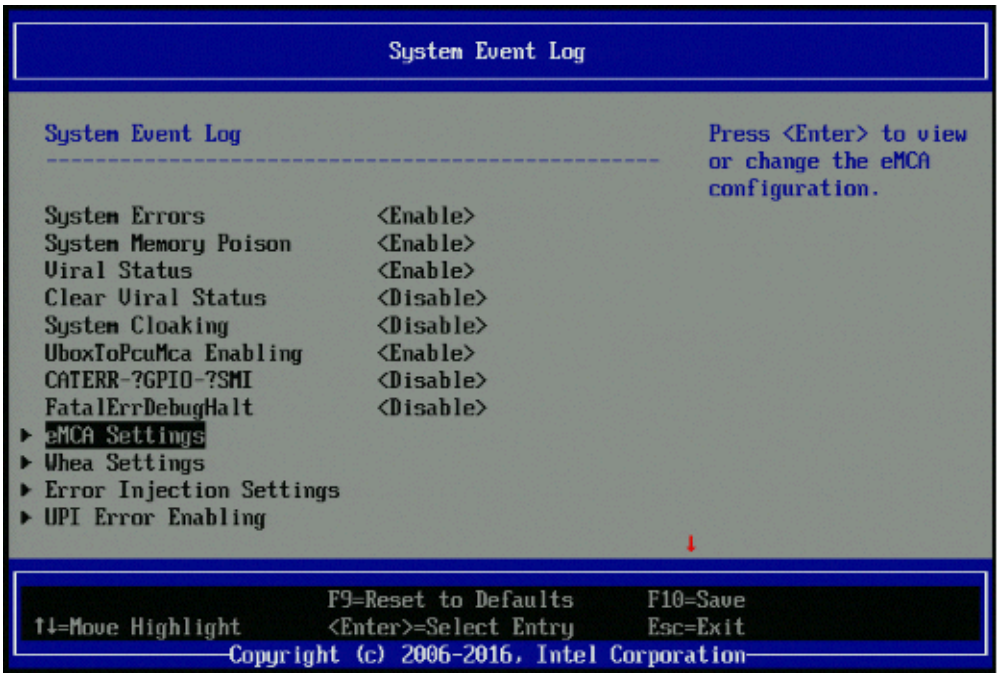
LMCE AND VIRTUALIZATION SUPPORT

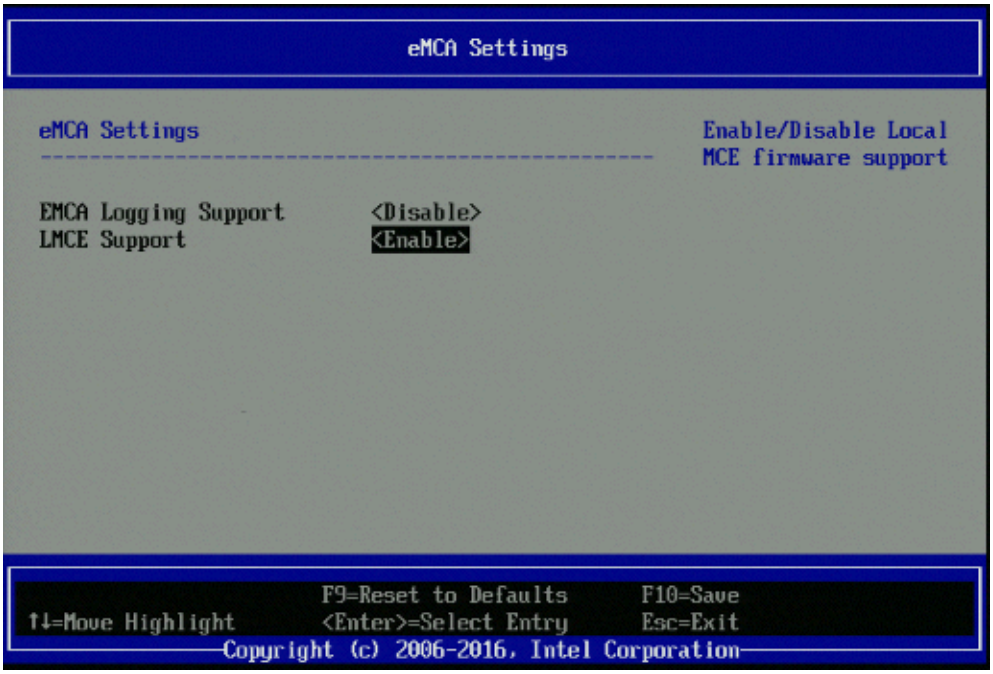
Linux KVM and Qemu are also enabled to benefit from Local Machine Check Exception. KVM has supported LMCA since v4.8 and Qemu version 2.7 enabled LMCE.

ENABLING LMCE IN BIOS

The following illustrations are based on early Purley-based systems. Consult your OEM/BIOS vendor documentation on how to enable LMCE for your specific platforms.







LINUX KERNEL COMMAND LINE OPTIONS

Local Machine Check Exception handling is now available in Linux* and on servers using the Intel® Xeon® processor family starting with the X65xx and X75xx generation. Local Machine Check Exception handling avoids some of the pitfalls with broadcast-based machine checks to significantly improve robustness in cloud and modern system architectures.

[i] http://www.cpushack.com/2017/01/28/stratus-servers-that-wont-quit-the-24-year-running-computer/